Noname manuscript No.
(will be inserted by the editor)

MINQ8 – general definite and bound constrained indefinite quadratic programming

Waltraud Huyer · **Arnold Neumaier**

the date of receipt and acceptance should be inserted later

Keywords Definite quadratic programming · Bound constrained indefinite quadratic programming · Dual program · Feasibility problem

Abstract We propose new algorithms for (i) the local optimization of bound constrained quadratic programs, (ii) the solution of general definite quadratic programs, and (iii) finding a point satisfying given linear equations and inequalities or a certificate of infeasibility. The algorithms are implemented in MATLAB and tested against state-of-the-art quadratic programming software.

1 Introduction

We consider the quadratic programming problem with bound constraints

$$\begin{array}{l} \min\ c^T x + \frac{1}{2} x^T G x \\ \text{s.t.}\quad x \in \mathbf{x}, \end{array} \tag{1}$$

where $c \in \mathbb{R}^n$, G is a symmetric $n \times n$ matrix, not necessarily semidefinite, $\mathbf{x} := [\underline{x}, \overline{x}] = \{x \in \mathbb{R}^n \mid \underline{x}_i \le x_i \le \overline{x}_i,\ i = 1, \ldots, n\}$ is a box in $\mathbb{R}^n$, and infinite bounds are allowed. Quadratic programming problems with simple bounds form an important class of nonlinear optimization problems, including in the definite case linear least squares with nonnegativity constraints and linearly constrained least distance problems (in the dual formulation), and in the indefinite case a number of combinatorial problems such as the maximum clique problem or the maximum cut problem (see, e.g., Floudas and Pardalos [4]). The global optimization problem is quite difficult in the indefinite case. Here we do not discuss the global solution of indefinite quadratic programs,

Waltraud Huyer
Fakultät für Mathematik, Universität Wien, Oskar-Morgenstern-Platz 1, 1090 Wien, Austria
E-mail: Waltraud.Huyer@univie.ac.at

Arnold Neumaier
Fakultät für Mathematik, Universität Wien, Oskar-Morgenstern-Platz 1, 1090 Wien, Austria
E-mail: Arnold.Neumaier@univie.ac.at

whose reliable solution needs quite different techniques (relaxation and branch and bound); see, e.g., several articles in the collection by Lee and Leyffer [17]. However, local methods for quadratic programs are also important in the global case since they are needed for finding good starting points for a global solver.

There are many optimization algorithms designed for finding local optima of quadratic programs under various conditions. A fairly comprehensive list of algorithms is maintained by Gould and Toint [9]. For strictly convex bound constrained quadratic programs, many algorithms are available. For a comparison of solvers for strictly convex bound constrained quadratic programs see Voglis and Lagaris [25]. We are interested in two more general situations: bound constrained quadratic programs without restriction (they may be strictly convex, rank-deficient or indefinite), and strictly convex quadratic programs with general linear constraints. Because of duality, the second problem class may be viewed as a special case of the first, hence is still significantly simpler than a general quadratic program. The number of solvers that exploit this additional structure is quite small; see Section 5 below.

MINQ5 [20] is a publicly available MATLAB program for bound constrained quadratic programming and strictly convex general quadratic programming, based on rank 1 modifications. It finds a local minimizer of the problem (1). If G is positive semidefinite, any local optimizer is global, so it finds (in theory, assuming exact arithmetic) the global optimum. The method combines coordinate searches and subspace minimization steps. The latter are safeguarded equality constrained QP steps, performed whenever the coordinate searches no longer change the active set. Rank 1 updates, in a format suited for both the dense and sparse case, are used to keep the linear algebra reasonably cheap.

Applications of MINQ5 to bound constrained quadratic programs include the calculation of the formation constants for copper(I)-chloride complexes [19], subpixel land cover mapping [3], an optimal control model of redundant muscles in step-tracking wrist movements [10], entropy estimation for high-dimensional finite-accuracy data [16], finding the best code to represent a speech signal [24], the calculation of a nonnegative sparsity induced similarity measure for cluster analysis of spam images [7], a method for calibrating the scores of biased reviewers [22], a model for the braking behavior of industrial robots [2], and finding the filter coefficients for a filtering technique to denoise the far-field pattern in the presence of noise [21]. Moreover, many nonlinear optimization techniques are based on solving auxiliary quadratic problems [6, 14, 18, 25, 26]. Kanzow and Petra [15] used MINQ5 to solve a trust-region subproblem in a scaled filter trust region method. Finally, MINQ5 is an integral part of the global optimization algorithm MCS [12] and the noisy optimization algorithm SNOBFIT [13].

In the framework of another optimization problem we encountered bound constrained quadratic programs where MINQ5 turned out to be very slow. This problem provided the impetus for developing a new quadratic programming algorithm, which we call MINQ8. In contrast to MINQ5, we do not use rank 1 updates but instead make direct factorizations. Moreover, since the subspaces for the subspace steps for MINQ5 sometimes seemed to be too large, we use smaller subspaces in MINQ8.

In Section 2 we give a rough outline of our algorithm MINQ8, and in Section 3 we elucidate its main subprograms in detail, including the modifications needed

for infinite bounds. Moreover, we discuss properties of the points obtained by the algorithm and prove convergence under the assumption of strict convexity and special settings of the parameters. We describe how MINQ8 can be applied to general positive definite quadratic programs in Section 4. Finally, in Section 5 we make an extensive comparison of MINQ8, MINQ5, the algorithms contained in the `quadprog` function of MATLAB, and `NewtonKKTqp` [1] on different test problems.

Notation. Let $A = (A_{i,j})_{1\le i\le m, 1\le j\le n} \in \mathbb{R}^{m\times n}$, $x = (x_1,\dots,x_n)^T \in \mathbb{R}^n$, I a subset of $\{1,\dots,m\}$, and J a subset of $\{1,\dots,n\}$. Then $A_{I,J}$ denotes the submatrix of A with the row indices taken from I and the column indices from J, i.e., $A_{I,J} := (A_{i,j})_{i\in I, j\in J}$, $A_{I,:}$ the submatrix $A_{I,:} := (A_{i,j})_{i\in I, 1\le j\le n}$, and similarly $A_{:,J} := (A_{i,j})_{1\le i\le m, j\in J}$. If $J = \{l\}$, we also write $A_{:,l}$ instead of $A_{:,J}$, analogously for the other cases of one-element index sets. x_J is the subvector of x with the indices taken from the set J, and the vector $|x| := (|x_1|,\dots,|x_n|)^T$ is obtained by taking the absolute values of all components of x.

2 The MINQ8 algorithm

MINQ8 is designed to solve the problem of finding a *local* minimizer of a bound constrained (definite or indefinite) quadratic problem of the form

$$\begin{array}{l} \min\ f(x) := \gamma + c^T x + \rho(Ax-b) \\ \text{s.t.}\quad x \in \mathbf{x}, \end{array} \tag{2}$$

where $\gamma\in\mathbb{R}$, $c\in\mathbb{R}^n$, $b\in\mathbb{R}^m$, $A\in\mathbb{R}^{m\times n}$, $\mathbf{x}$ is as before, and $\rho:\mathbb{R}^m\to\mathbb{R}$ is given by

$$\rho(z) := \frac{1}{2} z^T D z,$$

where $D\in\mathbb{R}^{m\times m}$ is a diagonal matrix. Thus there are n variables and each residual $Ax-b$ has m components. Note that this form of writing a quadratic function $\gamma + c^T x + \frac{1}{2}x^T G x$ is not a restriction since each symmetric matrix $G\in\mathbb{R}^{n\times n}$ can be written as $G = A^T D A$, $A, D\in\mathbb{R}^{n\times n}$, D a diagonal matrix, using a spectral factorization or an LDL^T factorization of G.

The algorithm performs four different subtasks, `fixbounds`, `reductionstep`, `subspacestep`, and `freebounds`. Each of these subprograms yields, starting from the point x produced by the previously called subprogram (or the initial point) with function value f, a point x^{new} with function value $f_{\text{new}} < f$ or stays at $x^{\text{new}} = x$, and x^{new} is used as input for the subsequent subtask. The subtasks are carried out in an appropriate sequence according to Algorithm 1. Here f is the function value at the current point x, f_{old} is the function value obtained before the previous call to `freebounds` (and initially it is set to ∞), and δ_1 is a fixed small positive number to handle the stopping criterion for function values close to zero. This number and other parameters in the algorithm will be specified in Section 5.

The subprogram `fixbounds` cyclically tries to fix as many variables at bounds as possible. The subprogram `reductionstep` reduces the set of predicted inactive variables if it contains more than m elements, by fixing some coordinates at a bound while leaving Ax constant. The subprogram `subspacestep` defines a search direction

Input: x, `maxit`, `tol`, problem characteristics
check whether the problem is unbounded along a coordinate and return in that case
check whether the function does not depend on some variables
for *nit* $\leftarrow 1$ **to** `maxit` **do**
  `fixbounds`
  compute the predicted inactive set
  if *the predicted inactive set is large* **then**
    `reductionstep`
  end
  `subspacestep`
  if `subspacestep` *has not changed the active set* **then**
    if $f = f_{old}$ **or** $(f_{old} - f)/\max(|f_{old}|, |f|, \delta_1) <$ `tol` **then**
      break
    end
    `freebounds`
  end
end
Output: x, f, `nit`

Algorithm 1: The MINQ8 algorithm

p, where p_K (with the predicted active set K) is the step to the predicted activities and p_I is the step to the unconstrained minimizer, and makes a line search along $x + \alpha p \in \mathbf{x}$. The subprogram `freebounds` allows the possibility of freeing bounds. The program stops due to finding an unbounded direction (before entering the loop or in the loop), reaching the maximum number `maxit` of iteration, or due to an insufficient change in the function value given by the input parameter `tol`. Since a very small step is rarely followed by a large step (as most ingredients of deterministic algorithms depend continuously on the current point), it is typical of optimization algorithms to terminate them when the gain in function value is too small. The algorithm returns the best point, its function value, the number of times the loop in Algorithm 1 has been carried out, and an error flag indicating whether the problem was detected to be unbounded.

Note that, due to the fact that the subtasks of the algorithm only accept a new point if it has a strictly smaller function value, a local minimizer returned might belong to a locally optimal ray. In particular, the algorithm checks at the beginning whether the objective function is constant along some coordinate axis; cf. the initial part of Section 3. Moreover, if the function value has not been changed by MINQ8, the current point has not changed. Therefore, in the case that `tol` is set to zero, the algorithm stops at a Kuhn–Tucker point; cf. Lemma 2.

3 Algorithmic details

In this section we describe the ingredients of the algorithm in detail. The following notation will be used in this section. Let

$$d_i := \frac{1}{2} A_{:,i}^T D A_{:,i}, \quad i = 1, \ldots, n. \tag{3}$$

Then

$$f(x + \alpha e^i) = f + \alpha g_i + \alpha^2 d_i =: f + \Delta f,$$

where e^i is the ith standard basis vector of $\mathbb{R}^n$,

$$g := c + A^T D(Ax - b) \tag{4}$$

denotes the gradient of f at x and $f := f(x)$ (by abuse of notation). The minimizer of $f(x+\alpha e^i)$, $x+\alpha e^i \in \mathbf{x}$, along the coordinate i is attained at one of

$$\alpha = \underline{\alpha}_i := \underline{x}_i - x_i, \quad \Delta f = \underline{\Delta} f_i := \underline{\alpha}_i (g_i + \underline{\alpha}_i d_i), \tag{5}$$

$$\alpha = \overline{\alpha}_i := \overline{x}_i - x_i, \quad \Delta f = \overline{\Delta} f_i := \overline{\alpha}_i (g_i + \overline{\alpha}_i d_i), \tag{6}$$

$$\alpha = \hat{\alpha}_i := -\frac{g_i}{2d_i}, \quad \Delta f = \hat{\Delta} f_i := \frac{g_i}{2}\hat{\alpha}_i = -\frac{g_i^2}{4d_i}, \tag{7}$$

where the third case only occurs if $\underline{\alpha}_i < \hat{\alpha}_i < \overline{\alpha}_i$ and $d_i > 0$. Note that the quantities d_i are independent of the point x and stay the same during the algorithm, but the gradient g has to be recomputed at each point. Thus, if $d_i < 0$ and $\underline{x}_i = -\infty$ or $\underline{x}_i = \infty$ for some i, the algorithm identifies the problem as unbounded at the beginning of the algorithm, terminates the execution and does not enter the loop in Algorithm 1; for details see Subsection 3.6. The same holds in the case that $d_i = 0$ and either $\underline{x}_i = -\infty$ if $g_i > 0$ or $\overline{x}_i = \infty$ if $g_i < 0$. Moreover, if $c_i = 0$ and $A_{:,i} = 0$ for some i, f does not depend on x_i. In that case, x_i is fixed at its initial value and only the remaining variables are optimized.

The **active set** of the current point x is

$$\{i \mid x_i = \underline{x}_i \text{ or } x_i = \overline{x}_i\};$$

the variables that are not active are called **free** (or **inactive**) **variables**.

The easiest case is that of an unconstrained convex quadratic optimization problem, where the global optimizer can be computed easily by satisfying the Kuhn–Tucker conditions. If $x \in \mathbb{R}^n$ with the corresponding gradient g and the Hessian matrix G, the global minimum is attained at $x+p$ with $Gp = -g$. However, in the nonconvex case even the problem of checking the existence of a Kuhn–Tucker point is NP-hard; cf. Horst et al. [11].

If the unconstrained minimizer of a convex quadratic optimization problem is not contained in $\mathbf{x}$ and in all other cases where the objective function is bounded on the box $\mathbf{x}$, minimizers occur at the boundary. In particular, if a concave quadratic function is minimized over a bounded box $\mathbf{x}$, then an optimal solution is attained at a vertex of $\mathbf{x}$. The global minimizer of an indefinite quadratic program does not necessarily occur at a vertex of $\mathbf{x}$ but at the boundary of the feasible domain. More generally, if the Hessian matrix G has s negative eigenvalues, then at least s bounds are active at the global minimizer; see Horst et al. [11], for example. Moreover, for the problem considered here, we have $\mathrm{rk}(G) \le \min(n,m)$ and each $k \times k$-submatrix of G with $k > m$ is singular. Therefore it makes sense to fix $n-k$ variables at the boundary and look for the optimal step for an unconstrained convex quadratic optimization problem in a k-dimensional subspace, $k \le m$.

Due to the discussion in the previous paragraph, it is important to try to find the correct activities, which is done in `fixbounds`. Looking for the correct activities is alternated with attempts to find the optimal values of the predicted nonactive variables (`subspacestep`). Nevertheless `subspacestep` (if necessary combined with a `reductionstep`), like `fixbounds`, can only cause an increase in the

number of active variables. If both `fixbounds` and `subspacestep` (with or without `reductionstep`) do not increase the number of active variables, `freebounds` is called, i.e. a line search along a direction p that is a descent direction in the special case that `subspacestep` has not changed the function value and thus the point; cf. Lemma 1.

In the following we describe the ingredients of MINQ8 in detail.

3.1 `fixbounds`

In the order of increasing d_i, try $\underline{\alpha}_i$, $\overline{\alpha}_i$ defined by (5) and (6), accept it if $\underline{\Delta} f_i < 0$ or $\overline{\Delta} f_i < 0$ (the smaller of both). This is repeated until no additional bound is fixed. Since the algorithm does not enter the loop and subsequently does not call `fixbounds` if $d_i < 0$ for some i and at least one of the corresponding bounds is infinite, `fixbounds` might only detect an unbounded direction for an i with $d_i = 0$ due to the change in g_i. In that case, the algorithm is terminated; otherwise `fixbounds` produces a point with a larger active set and a smaller function value (or it stays at the same point)

3.2 The predicted inactive set

For the point x obtained by `fixbounds`, we determine the **predicted inactive set** (a subset of the inactive set of x) and the **predicted active set** (a superset of the active set of x) as follows. For a fixed $\kappa \geq 0$ and $i = 1, \ldots, n$, we compute $\Delta f_i := \min(|\underline{\Delta} f_i|, |\overline{\Delta} f_i|, \kappa |\hat{\Delta} f_i|)$, where $\underline{\Delta} f_i$, $\overline{\Delta} f_i$, and $\hat{\Delta} f_i$ are defined by (5)–(7). Then the predicted inactive set is $I := \{i \mid \Delta f_i = \kappa |\hat{\Delta} f_i| < \min(|\underline{\Delta} f_i|, |\overline{\Delta} f_i|)\}$, the predicted active set is its complement $K := \{1, \ldots, n\} \setminus I$, and Δf_i is finite for $i = 1, \ldots, n$. Since x is the output of `fixbounds`, i.e., it is not possible to make a gain in function value by fixing an additional bound, we have $\underline{\Delta} f_i \geq 0$ and $\overline{\Delta} f_i \geq 0$. If $x_i \in \{\underline{x}_i, \overline{x}_i\}$, i.e., i is an active index, we have $\Delta f_i = 0$ and i is also a predicted active index. If i is an inactive index, $\underline{\Delta} f_i$ and $\overline{\Delta} f_i$ are nonnegative and $\hat{\Delta} f_i \leq 0$. Then, even in the case that $\hat{\Delta} f_i < \min(\underline{\Delta} f_i, \overline{\Delta} f_i)$, i belongs to the predicted active set if $\min(\underline{\Delta} f_i, \overline{\Delta} f_i) \leq -\kappa \hat{\Delta} f_i$, i.e., if the loss in function value by setting x_i to a boundary value is not too large compared to the gain obtained by setting x_i to its optimal value. In the special case that $\kappa = 0$, the predicted active and inactive sets coincide with the active and inactive sets, respectively. Another special case is $\kappa = 1$, where $|\underline{\Delta} f_i|$, $|\overline{\Delta} f_i|$, and $|\hat{\Delta} f_i|$ are compared.

3.3 `reductionstep`

Let I be the predicted inactive set of the current point x. If $|I| > m$, $G_{I,I}$ is singular and we reduce the number of elements of I by the following procedure to obtain $|I| \leq m$.

The idea is to first perform an approximate minimization on the subspace $Ax = Ax^0$, where x^0 is the current best point. On this subspace, the objective function sim-

plifies to $c^T x$ plus a constant, and all currently active variables are fixed. Thus proceeding along a descent direction (if there is one) of the linear program

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t.}\ \ Ax = Ax^0, \quad x \in \mathbf{x} \end{aligned}$$

we are guaranteed to fix a bound, and this can be iterated in strongly polynomial time until $|I| \le m$.

Noting that the case $|I| > m$ is only possible for $n > m$, we use LU factorizations of $A^T_{:,I}$ and submatrices of $A^T_{:,I}$ to determine maximal subsets I' of I and J' of $\{1,\dots,m\}$, $m \ge r := |I'| = |J'|$, such that $A_{J',I'}$ is invertible, and compute its inverse $C := A^{-1}_{J',I'}$. The set $J'' := \{1,\dots,m\} \setminus J'$ might be empty, but $I'' := I \setminus I' \ne \emptyset$ since $|I| > m$.

We now define a vector $u \in \mathbb{R}^n$ with $Au = 0$ as follows. Let $u_{I''} = e^k$ for some $k \in \{1,\dots,|I''|\}$, and let $u_{I'} := -CA_{J',I''}u_{I''}$ and $u_K = 0$. Then we have $A_{:,I''}u_{I''} = A_{:,l}$ for some $l \in I''$ and

$$Au = A_{:,I'}u_{I'} + A_{:,I''}u_{I''} = -A_{:,I'}CA_{J',I''}u_{I''} + A_{:,l} = -A_{:,I'}CA_{J',l} + A_{:,l}.$$

Since r is the (numerical) rank of $A_{:,I}$ and $A_{J',I'}$ is nonsingular, the columns of $A_{:,I'}$ form a basis of the column space of $A_{:,I}$. Hence $A_{:,l}$ can be written as a linear combination

$$A_{:,l} = \sum_{j=1}^{r} \lambda_j A_{:,i_j}$$

for appropriate λ_j, where $I' = \{i_1,\dots,i_r\}$. Then

$$\begin{aligned} Au &= -A_{:,I'}CA_{J',l} + A_{:,l} = -\textstyle\sum_{j=1}^{r} \lambda_j A_{:,I'}CA_{J',i_j} + A_{:,l} \\ &= -\textstyle\sum_{j=1}^{r} \lambda_j A_{:,I'}e^j + A_{:,l} = -\sum_{j=1}^{r} \lambda_j A_{:,i_j} + A_{:,l} = 0, \end{aligned}$$

where we have used the definition of C.

Since $A(x+\alpha u) = Ax$, minimizing $c^T(x+\alpha u) + \rho(A(x+\alpha u))$ over the box $\mathbf{x}$ amounts to minimizing $\alpha c^T u$ subject to $x + \alpha u \in \mathbf{x}$, which is easy to do.

Putting things together results in Algorithm 2. $|I''|$ choices for $u_{I''}$ and thus for u are possible, but in the first `for` loop the algorithm just looks for one u that manages to fix at least one bound, and that will usually be the one with $u_{I''} = e^1 \in \mathbb{R}^{|I''|}$ since the standard basis vectors are tried out in the natural order. Let $L \subset I$ be the set of bounds that are fixed. In the second `for` loop the elements $i \in L$ are transferred from I to K. If $i \in I''$, this can be done directly, thus reducing I''. In the case where $i \in I'$, $|I'|$ should stay the same. We therefore replace i by a $j \in I''$ for which $A_{J',\bar{I}'}$ is still invertible, where $\bar{I}' := (I' \cup \{j\}) \setminus \{i\}$, and $\bar{C} := A^{-1}_{J',\bar{I}'}$ is computed from C with the aid of the Sherman–Morrison formula. The index j is chosen such that the denominator in the Sherman–Morrison formula is maximal to improve numerical stability. This procedure is repeated until $I = I'$, $I'' = \emptyset$. The indices that are transferred from I to K result in active indices of the new current point. The matrix $A_{:,I}$ obtained at the end of the algorithm has full rank by construction, which is necessary (but not sufficient) for $G_{I,I} = A^T_{:,I}DA_{:,I}$ to be nonsingular.

Input: x, I, I', J', problem characteristics
$K \leftarrow \{1,\dots,n\} \setminus I$
$I'' \leftarrow I \setminus I'$
while $I'' \neq \emptyset$ **do**
  for *all possible choices for* u **do**
    Go with α to the boundary that minimizes $c^T(x+\alpha u)$
    $L \leftarrow \{i \in I \mid x_i + \alpha u_i = \underline{x}_i \text{ or } \overline{x}_i\}$
    if $L \neq \emptyset$ **then**
      break
    end
  end
  $x \leftarrow x + \alpha u$
  for $i \in L$ **do**
    if $i \in I'$ **then**
      swap i with $j \in I''$ such that $|1+(Ch)_{i'}|$, $h := A_{J',j} - A_{J',i}$, is maximal and i' is chosen such that i is the i'th element of I'
      $I' \leftarrow (I' \setminus \{i\}) \cup \{j\}$, $I'' \leftarrow (I'' \setminus \{j\}) \cup \{i\}$
      $C \leftarrow (A_{J',I'} + h(e^{i'})^T)^{-1} = C - \frac{(Ch)C_{i',:}}{1+(Ch)_{i'}}$
    end
    $I'' \leftarrow I'' \setminus \{i\}$, $I \leftarrow I \setminus \{i\}$, $K \leftarrow K \cup \{i\}$
  end
end
Output: x, I, K

Algorithm 2: `reductionstep`

3.4 `subspacestep`

Now we have a predicted inactive set I with $|I| \leq m$ and a predicted active set K, which is a superset of the active set K' of the current point x with the gradient g and the Hessian matrix G. Let p_K be the step to the predicted activities determined as in Subsection 3.2 and by the additional activities generated by `reductionstep`; then $p_{K'} = 0$. The optimal p_I is then determined by $Gp = -g$, which results in solving

$$G_{I,I} p_I = -g_I - G_{I,K} p_K. \tag{8}$$

To improve numerical stability we regularize the matrix G by slightly increasing its diagonal entries. We set $G_{i,i} = (1+\delta_2)G_{i,i}$ if $G_{i,i} \neq 0$ and $G_{i,i} = \delta_2$ if $G_{i,i} = 0$, where δ_2 is a fixed small nonnegative number. In the case $\delta_2 = 0$ G remains unchanged. The vector p_I is computed from (8) using the backslash operator in MATLAB. $p_I = -G_{I,I} \backslash h_I$, which yields a least squares solution in the case that $G_{I,I}$ is singular. Subsequently an exact line search is made to find the best α with $x + \alpha p \in \mathbf{x}$, which is not necessarily $\alpha = 1$ but can even be negative, and a new current point $x + \alpha p$ is obtained. However, since $p_i = 0$ for active indices i, we always have $x + \alpha p \in \mathbf{x}$ at least for small $|\alpha|$. $p = 0$ is only possible if $g_I = 0$ and all indices in K are active.

3.5 `freebounds`

Here $f(x+\alpha p)$ is minimized for $x + \alpha p \in \mathbf{x}$ with $p_i = \alpha_i$, where $\alpha_i \in \{\underline{\alpha}_i, \overline{\alpha}_i, \hat{\alpha}_i\}$ with smallest Δf as defined by (5)–(7). This choice of p would be optimal for separable

quadratic problems. If an unbounded direction is detected in that procedure (for a coordinate i with $d_i = 0$ due to a change in g_i), the algorithm is terminated.

Lemma 1 *If* `freebounds` *is applied to the output of* `fixbounds` *(which is the case if* `reductionstep` *and* `subspacestep` *do not change the point), then p is a descent direction or $p = 0$.*

Proof Let x be the output of `fixbounds` and $i \in \{1,\dots,n\}$. If $\underline{x}_i < x_i < \overline{x}_i$, then for $\underline{\alpha}_i$, $\overline{\alpha}_i$, $\underline{\Delta} f_i$, and $\overline{\Delta} f_i$ defined by (5) and (6) we have $\underline{\alpha}_i < 0$, $\overline{\alpha}_i > 0$, $\underline{\Delta} f_i \geq 0$ and $\overline{\Delta} f_i \geq 0$. This implies $-\overline{\alpha}_i d_i \leq g_i \leq -\underline{\alpha}_i d_i$ and thus $d_i \geq 0$. If $d_i = 0$, we also have $g_i = 0$ and $p_i = 0$. For $d_i > 0$ we obtain $\frac{1}{2}\underline{\alpha}_i \leq -\frac{g_i}{2d_i} \leq \frac{1}{2}\overline{\alpha}_i$. Then for $\hat{\alpha}_i$ and $\hat{\Delta} f_i$ defined by (7) we have $\underline{\alpha}_i < \hat{\alpha}_i < \overline{\alpha}_i$ and $\hat{\Delta} f_i < 0$, which implies $p_i = \hat{\alpha}_i = -\frac{g_i}{2d_i}$.

If $x_i = \underline{x}_i$, then $\underline{\alpha}_i = 0$ and $\overline{\alpha}_i > 0$. Then $d_i \leq 0$ implies $p_i = \underline{\alpha}_i = 0$. $p_i = -\frac{g_i}{2d_i}$ is only possible in the case that $d_i > 0$ and $0 < -\frac{g_i}{2d_i} < \overline{\alpha}_i$, which yields $g_i < 0$. Similarly, if $x_i = \overline{x}_i$, we either have $p_i = 0$ or $p_i = -\frac{g_i}{2d_i}$, $d_i > 0$, and $g_i > 0$.

If $p \neq 0$, the above discussion shows that $x + \alpha p \in [\underline{x}, \overline{x}]$ for sufficiently small positive α and $p^T g = -\sum_{i \in M} \frac{g_i^2}{2d_i} < 0$, where $M := \{i \mid p_i \neq 0\}$.

3.6 Infinite bounds

Infinite bounds need a special treatment since floating-point arithmetic fails to work properly for numbers with a high absolute value. Let I_l be the set of i such that $\underline{x}_i = -\infty$, let I_u be the set of i such that $\overline{x}_i = \infty$, and assume that $I_{\text{inf}} := I_l \cup I_u \neq \emptyset$. If there is an $i \in I_{\text{inf}}$ such that $d_i < 0$, an $i \in I_l$ such that $d_i = 0$ and $g_i > 0$ or an $i \in I_u$ such that $d_i = 0$ and $g_i < 0$, the function is unbounded below on $\mathbf{x}$. The program returns an error flag and a finite vector x with $f(x) \ll 0$ still finite. The same is done in the case that an unbounded direction is found in `freebounds` or in an exact line search.

Since $|I| > m$ can only occur in the case $n > m$, where the Hessian matrix is singular, the algorithm in Subsection 3.3 either finds an infinite descent direction or manages to reduce $|I|$. The only exception is the case where $|c^T u|$ is small, i.e. the search direction leaves $c^T x$ (almost) constant. In the case that $|c^T u| \leq \delta_3 |c|^T |u|$ (where δ_3 is a fixed small positive number) and the minimization would yield an infinite descent direction, we therefore do not minimize $\alpha c^T u$ but set $\alpha = 0$; in the case $c^T u = 0$ (the function is constant along the direction u) we also set $\alpha = 0$. If that is the case for all possible choices of u, it is not possible to reduce I and `subspacestep` is carried out with a larger I.

3.7 Termination and convergence

Since all steps in MINQ8 are constructed such that either the point stays the same or the function value of the new point is strictly smaller, no improvement in function value implies no change in the current point. If the algorithm stops with no change in the function value, the following holds.

Lemma 2 *If a point x is not changed by* `fixbounds`, `reductionstep`, `subspacestep`, *and* `freebounds`, *then x is a Kuhn–Tucker point, i.e., the reduced gradient g^{red} at x, defined by*

$$g_i^{red} := \begin{cases} g_i & \text{if } \underline{x}_i < x_i < \overline{x}_i, \\ \max(0,-g_i) & \text{if } x_i = \underline{x}_i, \\ \max(0,g_i) & \text{if } x_i = \overline{x}_i, \end{cases} \tag{9}$$

where g is the gradient of of f at x, is zero.

Proof Let x be point that `fixbounds`, `reductionstep`, `subspacestep`, and `freebounds` leave the same. Then, by Lemma 1, the vector p defined by `freebounds` should be 0. Let $i \in \{1,\dots,n\}$. If $\underline{x}_i < x_i < \overline{x}_i$, $p_i = 0$ implies $g_i = 0$. Let now $x_i = \underline{x}_i$, i.e. $0 = \underline{\alpha}_i < \overline{\alpha}_i$ with the definitions from (5) and (6). In the case $d_i \le 0$, $\overline{\Delta} f_i = \overline{\alpha}_i(g_i + \overline{\alpha}_i d_i) \ge 0$ yields $g_i \ge -\overline{\alpha}_i d_i \ge 0$. If $d_i > 0$, we have $-\frac{g_i^2}{4d_i} \le 0$, and $p_i = 0$ implies $\hat{\alpha}_i = -\frac{g_i}{2d_i} \le \overline{\alpha}_i = 0$, again yielding $g_i \ge 0$. Similarly $x_i = \overline{x}_i$ implies $g_i \le 0$.

Under certain conditions, the result of `subspacestep` only depends on the activities.

Lemma 3 *Assume that the parameters κ in Subsection 3.2 and δ_2 in Subsection 3.4 are set to 0 and $x \in \mathbf{x} := [\underline{x},\overline{x}]$. If the subset I of $\{1,\dots,n\}$ is such that $G_{I,I}$ is positive definite, $K = \{1,\dots,n\} \setminus I = K_l \cup K_u$, $K_l := \{i \in \{1,\dots,n\} \mid x_i = \underline{x}_i\}$, $K_u := \{i \in \{1,\dots,n\} \mid x_i = \overline{x}_i\}$, and $G_{I,I}^{-1}(A_{:,I}^T Db - c_I - G_{I,K}x_K) \in [\underline{x}_I,\overline{x}_I]$, then the point obtained from x with the inactive set I by applying* `subspacestep` *only depends on K_l and K_u.*

Proof Since K_l and K_u are given, x_K is uniquely determined. $\kappa = 0$ implies that the predicted inactive set is equal to the inactive set I of x and thus the search direction p defined by `subspacestep` fulfils $p_K = 0$ for the active set $K = K_l \cup K_u$ and $G_{I,I}p_I = -g_I$, which by assumption yields the unique solution $p_I = -G_{I,I}^{-1}g_I$. Then $x_I + p_I = x_I - G_{I,I}^{-1}(c_I + G_{I,:}x - A_{:,I}^T Db) = -G_{I,I}^{-1}(c_I + G_{I,K}x_K - A_{:,I}^T Db)$, which is independent of x_I and in $[\underline{x}_I,\overline{x}_I]$ by assumption. The line search along $x + \alpha p$ yields $\alpha^* = 1$ since $x + p$ is the minimizer of $f(y)$ s.t. $y_K = x_K$, $y_I \in [\underline{x}_I,\overline{x}_I]$.

Corollary 1 *Assume that the parameters κ in Subsection 3.2 and δ_2 in Subsection 3.4 are set to 0 and that G is positive definite. If a point x with the correct activities has been found by* `fixbounds`, *then* `subspacestep` *yields the unique minimizer.*

Since both `fixbounds` and `subspacestep` cannot free any bounds and the number of active variables cannot increase indefinitely, it has to happen from time to time that `subspacestep` does not change the activities and `freebounds` is called.

Theorem 1 *Let the assumptions of Corollary 1 be satisfied. Let x^k be the current point before the kth call to* `freebounds`. *Then (x_k) cannot be an infinite sequence, i.e., the algorithm stops after finitely many iterations even if* `tol` *is set to 0 and the limit on the number of iterations to ∞.*

Proof If $x^k = x^{k+1}$, the stopping criterion in Algorithm 1 regarding no change in function value is fulfilled, and $x^k \neq x^l$ for $|k-l| > l$ since the function values form a decreasing sequence. So we are done if we show that there are only finitely many possible values of x^k.

Let x be the current point before the application of `freebounds`, I its inactive set, K its active set and g the corresponding gradient. If `subspacestep` has not changed the active set, two cases are possible. In the first case we have $p = 0$, i.e. the search direction of `subspacestep` is zero. Then x is the current point before and after the application of `subspacestep` and therefore $g_I = c_I + G_{I,:}x - A_{:,I}^T Db = 0$. Since G is nonsingular by assumption, we obtain $x_I = G_{I,I}^{-1}(A_{:,I}^T Db - c_I - G_{I,K}x_K)$, i.e., x_I is uniquely determined by the activities x_K.

In the second case, the line search into the direction of the minimizer of $f(y)$ s.t. $y_A = x_A$ yields a point x with x_I in the interior of $[\underline{x}_I, \bar{x}_I]$, which implies that x_I is the optimal point in the subspace, i.e., again $x_I = G_{I,I}^{-1}(A_{:,I}^T Db - c_I - G_{I,K}x_K)$. Therefore, in both cases, only a single choice of x_I is possible for every set of activities, and since there are only finitely many possible choices for the activities, the proof is finished.

4 Solving strictly convex quadratic programs

We consider the strictly convex quadratic program subject to linear constraints

$$\begin{array}{l} \min\ f(x) := c^T x + \frac{1}{2} x^T G x \\ \text{s.t.}\quad (Ax)_I \geq b_I, \quad (Ax)_E = b_E, \end{array} \tag{10}$$

where $c \in \mathbb{R}^n$, G is a symmetric, positive definite $n \times n$ matrix, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and (I,E) is a partition of $\{1,\ldots,m\}$ with $|E| \leq n$.

Theorem 2 *Let x be a feasible point of* (10) *and let y be a feasible point of the bound constrained quadratic program*

$$\begin{array}{l} \min\ d(y) := \frac{1}{2}(A^T y - c)^T G^{-1}(A^T y - c) - b^T y \\ \textit{s.t.}\quad y_I \geq 0. \end{array} \tag{11}$$

Then

$$f(x) + d(y) \geq 0,$$

with equality iff x and y are optimal solutions of (10) *and* (11)*, respectively. In this case,*

$$Gx + c = A^T y, \tag{12}$$

and the complementarity conditions

$$(Ax - b)_i y_i = 0 \quad \textit{for all } i \in I \tag{13}$$

hold.

Proof Write $r := A^T y - c - Gx$. Then

$$\begin{aligned} f(x) + d(y) &= c^T x + \tfrac{1}{2} x^T G x + \tfrac{1}{2}(r+Gx)^T G^{-1}(r+Gx) - b^T y \\ &= (c+Gx)^T x + \tfrac{1}{2} r^T G^{-1} r + r^T x - b^T y = \tfrac{1}{2} r^T G^{-1} r + y^T (Ax-b) \\ &\geq y^T (Ax-b) = \sum_{i \in I} (Ax-b)_i y_i \geq 0. \end{aligned}$$

Equality is possible only if $r = 0$ and $(Ax-b)_i y_i = 0$ for all $i \in I$. □

Together with the feasibility constraints, (12) and (13) are the common primal-dual optimality conditions for (10) and (11). Thus these problems are dual to each other, and we may find the solution of (10) by first solving (11) with a bound constrained solver and then computing

$$x = G^{-1}(A^T y - c).$$

This is how MINQ8 solves strictly convex problems of the form (10).

Since G is positive definite, (10) has a unique solution if it is feasible. However, (11) is not necessarily strictly convex, and may have a direction p of infinite descent. Since

$$d(y+\alpha p) = d(y) + \alpha \big((A^T y - c)^T G^{-1} A^T - b^T \big) p + \frac{1}{2} \alpha^2 (A^T p)^T G^{-1} A^T p,$$

this is possible only if

$$A^T p = 0, \quad b^T p > 0, \quad p_i \geq 0 \quad \text{for all } i \in I. \tag{14}$$

But in this case, any feasible x satisfies

$$0 > (Ax-b)^T p = \sum_{i \in I} (Ax-b)_i p_i \geq 0,$$

contradiction. Thus p defines a certificate of infeasibility for (10). (Numerically, due to rounding errors, one may find in place of a direction of infinite descent just a huge dual solution y with a very large negative objective function value. In this case, y is itself an approximate certificate of infeasibility, and one may check whether $p = y$ satisfies (14) to sufficient accuracy to report infeasibility.)

In general, to obtain the form (2) required by MINQ8 one has to make a spectral factorization or an LDL^T factorization of G (cf. Section 2). If the sparsity pattern of G is not unfavorable, this can be done in very large dimensions.

However there are two special cases that deserve a separate handling.

(i) In the special case where G is a diagonal matrix, the problem (11) has already the form required for MINQ8.

(ii) Considering the even more specialized case where $c = 0$ and $G = I_n$ in (10), we see that bound constrained optimization can also be used to find either the point with minimal Euclidean norm satisfying a system of linear equations and inequalities, or a certificate of infeasibility of this system.

5 Numerical examples

In analogy to the MINQ5 package [20] (named after MATLAB 5), we call our new algorithm MINQ8. Our MINQ8 implementation solves both the general bound constrained quadratic program and the related problems discussed in Section 4.

The numerical examples were carried out with MATLAB 8.3.0.532 (R2014a) for Linux on an Intel Core i5-4670S processor with 3.1 GHz. The default value for `tol` is $\texttt{tol} = 10^{-8}$. The small parameters δ_1, δ_2, and δ_3 from Section 2 and Subsections 3.4, and 3.6 were set to $\delta_1 = 10^{-4}$, $\delta_2 = 10^{-10}$, and $\delta_3 = 10^{-12}$, respectively. Moreover, κ in Subsection 3.2 was set to $\kappa = 0.5$, which turned out to perform much better in practice than $\kappa = 0$ (the predicted active set coincides with the active set) and slightly better than $\kappa = 1$, although the convergence proofs in Subsection 3.7 assume $\delta_2 = \kappa = 0$.

In Subsections 5.1 and 5.2 we consider problems of the form (2), and Subsection 5.3 is devoted to problems of the form (10). On the first two test sets we compare MINQ8 with the default value for `tol` with MINQ5, `NewtonKKTqp`, and the MATLAB function `quadprog`. For all problems and algorithms (except for the interior-point-convex `quadprog` algorithm, which does not take a starting point), the zero vector is used as the starting point. If MINQ5 returns a nonzero error flag `ier` (if the maximal number iterations $\texttt{maxit} = 10n$ is exceeded or the algorithm incorrectly assumes that the problem is unbounded below), another call to MINQ5 is made with the result from the previous call as the starting point.

The MATLAB version `NewtonKKTqp` of the Newton-KKT interior method for indefinite quadratic programming by Absil and Tits [1] returns a local minimizer of the indefinite quadratic function $f(x) = \frac{1}{2}x^T Hx + c^T x$ subject to the linear inequality constraints $Ax \leq b$, starting with a feasible initial point. We implemented the bound constraints as $Ax \leq b$, and in the case of an unconstrained problem, we set A to the $2 \times n$ zero matrix and $b := (1,1)^T$ since the program exited with an error message if A was chosen to be empty or a one-row matrix.

The MATLAB function `quadprog` contains three algorithms for quadratic programming: interior-point-convex, trust-region-reflective, and active-set. The interior-point-convex algorithm is the default algorithm in MATLAB 8, but it handles only convex problems and does not allow the choice of an initial point. The trust-region-reflective algorithm is not applicable to unconstrained problems. We used an increased limit of 10000 (instead of the default value 200) on the number of iterations. Since in most of the cases only one of the `quadprog` algorithms did well and the other ones performed badly or were not applicable, we only report the result of a single `quadprog` algorithm (the one with the best performance for the problem class) for each function.

In Table 1 listing the problem characteristics, an empty entry means that the quantity is the same as in the previous line. In the tables with the results (Tables 2 and 4), the best function value f_{best} and the CPU time needed for executing the algorithm (excluding the time for preparing the input data) are given for all four algorithms. In addition, we present the number `nit` of iterations for MINQ8 (the number of times the loop in Algorithm 1 has been executed), for `NewtonKKTqp` (where it is called `nb_iter`) and for `quadprog`, and the number `nsub` of subspace steps for MINQ5.

An asterisk after the time indicates that `NewtonKKTqp` terminated with the warning "Computed a non-KKT stationary point".

5.1 Test set 1: Random sparse problems

We consider $A \in \mathbb{R}^{m\times n}$ sparse with 6 nonzero integer entries in $[-5,5]$ per row, $c_i, b_j \in \{0,1,2,3,4,5\}$, $i = 1,\dots,n$, $j = 1,\dots,m$, $\gamma = 0$, and $|D_{i,i}| \in \{1,2,3,4,5\}$, $i = 1,\dots,n$, where all entries are chosen randomly from uniform distributions. A was always chosen to have full rank. The same problem is solved without bounds (only in the positive definite case) or on the box $[-10,10]^n$, and the problems are modified by taking $D = D_1$, D_1 a positive definite diagonal matrix (i.e. $D_{i,i} \in \{1,2,3,4,5\}$), $D = -D_1$ and $D = ED_1$, where E is a diagonal matrix with diagonal elements randomly chosen from $\{-1,1\}$. The problem characteristics n, m, the signs of $D_{i,i}$ and the bounds are displayed in Table 1. This table also contains the number of activities `nact` of the best solution found among the algorithms. If the lowest f_{best} (up to 5 significant digits) was also achieved by MINQ8, `nact` is the number of exact activities (since MINQ8 easily finds exact inactivities due to `fixbounds`), and MINQ5 also finds exact inactivities for Problem 1c. In contrast, `NewtonKKT` very rarely yields exact inactivities. If the lowest function value was achieved by `NewtonKKT`, we applied `MINQ8` to the point obtained by `NewtonKKT`, which resulted in a point with a slightly lower function value, and we display the number of activities of that point. Note that in the case of concave problems with a singular Hessian (7–10b) it is possible for minimizers to occur also at points that are not vertices.

The results are presented in Table 2. For `quadprog`, we report the results obtained by the interior-point-convex algorithm for the convex (i.e. positive semidefinite) problems and the ones for the trust-region-reflective algorithm for the non-convex problems. The trust-region-reflective algorithm does not accept unconstrained problems and also does not find the global minimum for six of the ten bound constrained convex problems. The active-set `quadprog` algorithm yields lower f_{best} values than the trust-region-reflective algorithm for some of the non-convex problems, but it is very slow and sometimes does not even finish within a limit of 10000 iterations.

Problem 1 without bounds and with a positive definite $G := A^T DA$ is the only problem where the first call to MINQ5 returns the not yet optimal function value $-1.5510 \cdot 10^5$ and the error flag for a problem that is unbounded below, but the optimal function value and a zero error flag are obtained after the second call to MINQ5. The fact that two calls to MINQ5 are made is indicated in the `nsub` column of Table 2, where two numbers, connected with a plus sign, are given.

All algorithms yield the unique optimal function value f_{best} (up to the number of digits presented here) for the six convex unconstrained problems (1–6a). Even though the bound constrained convex problems (1–6b and 7–10a) also possess a unique local and therefore also global minimum f_{best}, MINQ8 and interior-point-convex `quadprog` are the only algorithms that always manage to find the optimal f_{best}. MINQ5 only finds the correct f_{best} in the cases where the minima coincide with the unconstrained minima (3–6b), i.e. no bounds are active at the best point, and `NewtonKKTqp` never achieves the correct f_{best} and sometimes, as explained at

No.	m	n	D	bounds	nact
1a	1000	1000	$+$	no bounds	
1b			$+$	$[-10,10]^n$	35
1c			$-$	$[-10,10]^n$	1000
1d			$\pm$	$[-10,10]^n$	888
2a	3000	3000	$+$	no bounds	
2b			$+$	$[-10,10]^n$	112
2c			$-$	$[-10,10]^n$	3000
2d			$\pm$	$[-10,10]^n$	2684
3a	1500	1000	$+$	no bounds	
3b			$+$	$[-10,10]^n$	0
3c			$-$	$[-10,10]^n$	1000
3d			$\pm$	$[-10,10]^n$	898
4a	4500	3000	$+$	no bounds	
4b			$+$	$[-10,10]^n$	1
4c			$-$	$[-10,10]^n$	3000
4d			$\pm$	$[-10,10]^n$	2714
5a	2000	1000	$+$	no bounds	
5b			$+$	$[-10,10]^n$	0
5c			$-$	$[-10,10]^n$	1000
5d			$\pm$	$[-10,10]^n$	921
6a	6000	3000	$+$	no bounds	
6b			$+$	$[-10,10]^n$	0
6c			$-$	$[-10,10]^n$	3000
6d			$\pm$	$[-10,10]^n$	2735
7a	1000	1500	$+$	$[-10,10]^n$	507
7b			$-$	$[-10,10]^n$	1498
7c			$\pm$	$[-10,10]^n$	1329
8a	3000	4500	$+$	$[-10,10]^n$	1525
8b			$-$	$[-10,10]^n$	4494
8c			$\pm$	$[-10,10]^n$	3974
9a	1000	2000	$+$	$[-10,10]^n$	994
9b			$-$	$[-10,10]^n$	1992
9c			$\pm$	$[-10,10]^n$	1629
10a	2500	5000	$+$	$[-10,10]^n$	2498
10b			$-$	$[-10,10]^n$	4981
10c			$\pm$	$[-10,10]^n$	4386

Table 1 Problem characteristics of Test Set 5.1

the beginning of this section, claims that it has found a non-KKT stationary point (which is actually a non-stationary point). So, in spite of using different parameter values than the ones in the convergence proof, MINQ8 does well on convex problems. Non-convex problems usually have several local minimizers, and MINQ8 also finds good f_{best} values for many non-convex problems. Even though `NewtonKKTqp` frequently achieves the lowest f_{best} for non-convex problems, they are not even local minima and the algorithm somehow gets stuck. Of course, MINQ8 might also fail (i.e. not find a local minimizer) if `tol` is too large or `maxstep` is too small. Since the performance of each algorithm is basically the same on each problem class of the problems reported here (and several more that are not reported), the random element in the problems does not seem to be crucial and we did not generate several problems with the same n and m.

	MINQ8			MINQ5			NewtonKKTqp			quadprog (ipc/trr)		
No.	nit	f_{best}	t[s]	nsub	f_{best}	t[s]	nit	f_{best}	t[s]	nit	f_{best}	t[s]
1a	2	$-1.5512\cdot 10^5$	0.2	$2+1$	$-1.5512\cdot 10^5$	5.8	2	$-1.5512\cdot 10^5$	0.5	1	$-1.5512\cdot 10^5$	0.3
1b	33	$-1.7493\cdot 10^3$	1.8	100	$-1.4030\cdot 10^3$	13.1	2	$2.7644\cdot 10^3$	3.1*	9	$-1.7493\cdot 10^3$	0.8
1c	2	$-2.5765\cdot 10^7$	0.1	0	$-2.6045\cdot 10^7$	0.2	131	$-2.5610\cdot 10^7$	186	23	$-2.2121\cdot 10^7$	0.2
1d	24	$-1.3744\cdot 10^7$	0.4	8	$-1.3099\cdot 10^7$	0.4	165	$-1.3580\cdot 10^7$	231*	27	$-1.3307\cdot 10^7$	0.2
2a	3	$-3.4473\cdot 10^6$	2.6	1	$-3.4473\cdot 10^6$	72	11	$-3.4473\cdot 10^6$	339	1	$-3.4473\cdot 10^6$	2.2
2b	76	$-5.3100\cdot 10^3$	60	133	$-5.3061\cdot 10^3$	143	2	$6.8150\cdot 10^3$	151*	12	$-5.3100\cdot 10^3$	16
2c	2	$-7.9075\cdot 10^7$	0.5	0	$-7.8353\cdot 10^7$	0.7	178	$-7.8482\cdot 10^7$	8719	24	$-6.5940\cdot 10^7$	0.5
2d	40	$-3.9264\cdot 10^7$	2.0	4	$-4.0065\cdot 10^7$	1.8	204	$-4.1036\cdot 10^7$	1010	35	$-3.8998\cdot 10^7$	0.8
3a	2	$6.2145\cdot 10^3$	0.2	1	$6.2145\cdot 10^3$	3.2	2	$6.2145\cdot 10^3$	0.5	1	$6.2145\cdot 10^3$	0.4
3b	2	$6.2145\cdot 10^3$	0.2	1	$6.2145\cdot 10^3$	3.2	2	$6.2891\cdot 10^3$	3.0*	4	$6.2145\cdot 10^3$	0.9
3c	2	$-3.4313\cdot 10^7$	0.2	0	$-3.4269\cdot 10^7$	0.2	119	$-3.4108\cdot 10^7$	165	1011	$-3.1461\cdot 10^7$	7.6
3d	6	$-1.6170\cdot 10^7$	0.2	6	$-1.5777\cdot 10^7$	0.5	187	$-1.6441\cdot 10^7$	261*	28	$-1.5474\cdot 10^7$	0.3
4a	2	$1.8621\cdot 10^4$	2.4	2	$1.8621\cdot 10^4$	8	2	$1.8621\cdot 10^4$	12	1	$1.8621\cdot 10^4$	3.1
4b	2	$1.8624\cdot 10^4$	2.3	2	$1.8624\cdot 10^4$	82	2	$1.8911\cdot 10^4$	85*	6	$1.8624\cdot 10^4$	9.4
4c	2	$-1.0289\cdot 10^8$	0.5	0	$-1.0287\cdot 10^8$	1.1	159	$-1.0208\cdot 10^8$	7534	16	$-8.5424\cdot 10^7$	0.6
4d	14	$-5.1231\cdot 10^7$	1.2	11	$-5.0978\cdot 10^7$	2.4	220	$-5.2754\cdot 10^7$	10530	35	$-4.9071\cdot 10^7$	0.9
5a	2	$1.2369\cdot 10^4$	0.2	1	$1.2369\cdot 10^4$	3.3	2	$1.2369\cdot 10^4$	0.5	1	$1.2369\cdot 10^4$	0.3
5b	2	$1.2369\cdot 10^4$	0.2	1	$1.2369\cdot 10^4$	3.2	2	$1.2405\cdot 10^4$	3.1*	4	$1.2369\cdot 10^4$	0.6
5c	2	$-4.2284\cdot 10^7$	0.1	0	$-4.1971\cdot 10^7$	0.3	107	$-4.1134\cdot 10^7$	151	18	$-3.3676\cdot 10^7$	0.2
5d	40	$-2.0292\cdot 10^7$	0.9	1	$-1.9308\cdot 10^7$	0.4	193	$-1.9701\cdot 10^7$	275	27	$-1.8812\cdot 10^7$	0.2
6a	2	$3.8614\cdot 10^4$	2.7	1	$3.8614\cdot 10^4$	83	2	$3.8614\cdot 10^4$	12	1	$3.8614\cdot 10^4$	3.0
6b	2	$3.8614\cdot 10^4$	2.5	1	$3.8614\cdot 10^4$	83	2	$3.8722\cdot 10^4$	84*	1	$3.8614\cdot 10^4$	12
6c	2	$-1.2808\cdot 10^8$	0.6	0	$-1.2721\cdot 10^8$	1.4	157	$-1.2316\cdot 10^8$	7550	20	$-1.0369\cdot 10^8$	0.8
6d	39	$-6.0433\cdot 10^7$	2.4	7	$-6.0547\cdot 10^7$	2.4	206	$-6.0894\cdot 10^7$	10057	44	$-5.7184\cdot 10^7$	1.5
7a	850	$-1.5373\cdot 10^4$	67	570	$-1.3517\cdot 10^4$	57	2	$3.2184\cdot 10^3$	18*	14	$-1.5373\cdot 10^4$	2.4
7b	2	$-2.8071\cdot 10^7$	0.2	0	$-2.8487\cdot 10^7$	0.3	169	$-2.8533\cdot 10^7$	931	26	$-2.3291\cdot 10^7$	0.3
7c	59	$-1.5816\cdot 10^7$	1.3	20	$-1.5516\cdot 10^7$	1.3	194	$-1.6388\cdot 10^7$	1073*	31	$-1.5226\cdot 10^7$	0.3
8a	2503	$-5.0900\cdot 10^4$	1738	1012	$-4.9023\cdot 10^4$	632	2	$3.6107\cdot 10^4$	542*	11	$-5.0900\cdot 10^4$	21
8b	2	$-9.1112\cdot 10^7$	0.5	0	$-9.1050\cdot 10^7$	1.2	281	$-9.0852\cdot 10^7$	45062	21	$-7.4709\cdot 10^7$	1.0
8c	145	$-5.0751\cdot 10^7$	9.4	21	$-4.9939\cdot 10^7$	3.6	237	$-5.1612\cdot 10^7$	35895	41	$-4.7165\cdot 10^7$	1.2
9a	1350	$-2.9951\cdot 10^4$	115	238	$-2.8128\cdot 10^4$	58	2	$8.8475\cdot 10^3$	44*	11	$-2.9951\cdot 10^4$	2.5
9b	2	$-3.1822\cdot 10^7$	0.2	0	$-3.1680\cdot 10^7$	0.5	200	$-3.2713\cdot 19^7$	2620	18	$-2.7303\cdot 10^7$	0.3
9c	52	$-1.8673\cdot 10^7$	1.6	155	$-1.8182\cdot 10^7$	5.3	230	$-1.8934\cdot 10^7$	2958	38	$-1.7580\cdot 10^7$	0.5
10a	3489	$-7.6119\cdot 10^4$	1674	857	$-7.3865\cdot 10^4$	411	2	$3.0464\cdot 10^4$	690*	12	$-7.6119\cdot 10^4$	19
10b	2	$-8.2895\cdot 10^7$	0.5	0	$-8.3253\cdot 10^7$	1.1	232	$-8.4774\cdot 10^7$	48471*	18	$-7.0562\cdot 10^7$	0.7
10c	126	$-4.6840\cdot 10^7$	9.3	134	$-4.4983\cdot 10^7$	11	249	$-4.7643\cdot 10^7$	51048	34	$-4.2740\cdot 10^7$	1.4

Table 2 Results for Test Set 5.1

We also want to show the behavior of MINQ8 for unbounded problems. Applying MINQ8 to the unbounded problem 7a yielded a warning "The problem is unbounded", a corresponding error flag and a finite point with function value $f = -5\cdot 10^{307}$ before entering the loop in Algorithm 1.

5.2 Test set 2: Quadratic problems from CUTEr

In addition, we coded some quadratic problems from the CUTEr test set [8] and the extended Rosenbrock function EROSEN from [23], defined by

$$f(x) = (1-x_1)^2 + \sum_{i=2}^{n} 100(x_i - x_{i-1})^2.$$

We use the default bounds and the default starting point for all problems. The names, dimensions, bounds, starting points, and numbers of activities at the global minimizers of the problems are listed in Table 3, and in order to save space in the presentation of the results, we number the problems from C1 to C19. For all problems we have $m \geq n$.

Table 4 contains the results. A failure of `NewtonKKTqp` is indicated by a dash: DQDRTIC terminated with an error message, the 100000-dimensional problems could not be solved with `NewtonKKTqp` due to limitations in memory, and CHENHARK produced a warning (`large k!!!`) and never finished. NCVXBQP1–3 and QUDLIN are the only non-convex problems in this test set. Since on this test set interior-point-convex `quadprog` occasionally fails to find the global minimum and most of the problems are convex, we report the results of interior-point-convex `quadprog` for all convex problems as well as the results of trust-region-reflective `quadprog` for all constrained problems.

No.	Name	n	m	bounds	starting point	nact
C1	BIGGSB1	1000	$n+1$	$[0,0.9]^n \times \mathbb{R}$	$(0,\ldots,0)$	999
C2	CHENHARK	1000	$n+2$	$[0,\infty)^n$	$(0.5,\ldots,0.5)$	499
C3	CVXBQP1	1000	n	$[0.1,10]^n$	$(0.5,\ldots,0.5)$	1000
C4		10000				10000
C5	DIXON3DQ	10000	n	no bounds	$(-1,\ldots,-1)$	
C6	DQDRTIC	5000	n	no bounds	$(3,\ldots,3)$	
C7	EROSEN	5000	n	no bounds	$(-1,\ldots,-1)$	
C8	HARKERP2	1000	$2n$	$[0,\infty)^n$	$(1,2,\ldots,n)$	999
C9	NCVXBQP1	1000	n	$[0.1,10]^n$	$(0.5,\ldots,0.5)$	1000
C10		10000				10000
C11	NCVXBQP2	1000	n	$[0.1,10]^n$	$(0.5,\ldots,0.5)$	993
C12		10000				9935
C13	NCVXBQP3	1000	n	$[0.1,10]^n$	$(0.5,\ldots,0.5)$	984
C14		10000				9839
C15	PENTDI	5000	$4n-6$	$[0,\infty)^n$	$(1,\ldots,1)$	4998
C16	QUDLIN	1200	$n+1$	$[0,10]^n$	$(1,\ldots,1)$	1200
C17		5000				5000
C18	TOINTQOR	50	83	no bounds	$(0,\ldots,0)$	
C19	TRIDIA	50	n	no bounds	$(1,\ldots,1)$	

Table 3 Problem characteristics of some CUTEr problems and EROSEN

The active-set `quadprog` algorithm was also tested, but it is inferior to the `quadprog` results presented in Table 4. It does not solve the 10000-dimensional problems due to memory limitations, and it takes hours to solve PENTDI and the 5000-dimensional QUDLIN problem.

	MINQ8			MINQ5			NewtonKKTqp		
No.	nit	f_{best}	t [s]	nsub	f_{best}	t [s]	nit	f_{best}	t [s]
C1	506	0.0150	5.8	502	0.0150	13	3	1.5	3.2*
C2	2	-2.0000	4.9	1	-2.0000	0.3	–	–	–
C3	2	$2.2523 \cdot 10^{4}$	0.1	0	$2.2523 \cdot 10^{4}$	0.1	13	$2.2523 \cdot 10^{4}$	19
C4	2	$2.2502 \cdot 10^{6}$	0.8	0	$2.2502 \cdot 10^{6}$	1.3	–	–	–
C5	3	$3.3750 \cdot 10^{-20}$	0.4	1	$4.4404 \cdot 10^{-10}$	15	–	–	–
C6	2	0	0.2	2	0	3.8	–	–	–
C7	3	$3.0119 \cdot 10^{-20}$	0.3	2	$2.2161 \cdot 10^{-8}$	4.3	2	$-4.4409 \cdot 10^{-16}$	18
C8	109	-0.5000	2.7	2	-0.5000	0.8	44	-0.5000	37
C9	2	$-1.9868 \cdot 10^{8}$	0.1	0	$-1.9868 \cdot 10^{8}$	0.1	35	$-1.9868 \cdot 10^{8}$	52
C10	2	$-1.9855 \cdot 10^{10}$	1.1	0	$-1.9855 \cdot 10^{10}$	1.2	–	–	–
C11	3	$-1.3339 \cdot 10^{8}$	0.1	2	$-1.3339 \cdot 10^{8}$	0.2	66	$-1.3339 \cdot 10^{8}$	97
C12	4	$-1.3340 \cdot 10^{10}$	1.4	2	$-1.3340 \cdot 10^{10}$	2.4	–	–	–
C13	5	$-6.5557 \cdot 10^{7}$	0.1	2	$-6.5776 \cdot 10^{7}$	0.2	66	$-6.5790 \cdot 10^{7}$	96
C14	10	$-6.5361 \cdot 10^{9}$	2.0	2	$-6.5584 \cdot 10^{9}$	2.8	–	–	–
C15	3	-0.7500	0.9	1	-0.7500	12	22	-0.7500	793
C16	2	$-7.2 \cdot 10^{7}$	0.2	0	$-7.2 \cdot 10^{7}$	0.2	2	$-8.0902 \cdot 10^{6}$	7.9*
C17	2	$-1.2500 \cdot 10^{9}$	5.7	0	$-1.2500 \cdot 10^{9}$	5.9	2	$-1.2882 \cdot 10^{8}$	668*
C18	2	$1.1755 \cdot 10^{3}$	0.1	1	$1.1755 \cdot 10^{3}$	0.1	2	$1.1755 \cdot 10^{3}$	0.1
C19	2	$3.3217 \cdot 10^{-37}$	0.1	1	$1.6021 \cdot 10^{-13}$	0.3	2	0	0.1

	quadprog (ipc)			quadprog (trr)		
No.	nit	f_{best}	t [s]	nit	f_{best}	t [s]
C1	8	0.0152	0.2	221	0.0150	2.9
C2	7	-2.0000	0.2	18	-2.0000	0.6
C3	8	$1.2523 \cdot 10^{4}$	0.3	13	$1.2523 \cdot 10^{4}$	0.3
C4	11	$1.2502 \cdot 10^{6}$	2.7	24	$1.2502 \cdot 10^{6}$	1.9
C5	1	0	1.2	–	–	–
C6	1	$1.2326 \cdot 10^{-30}$	0.4	–	–	–
C7	1	0	0.5	–	–	–
C8	14	-0.4912	14	24	-0.5000	1.0
C9	–	–	–	14	$-1.9868 \cdot 10^{8}$	0.2
C10	–	–	–	21	$-1.9855 \cdot 10^{10}$	2.0
C11	–	–	–	18	$-1.3339 \cdot 10^{8}$	0.1
C12	–	–	–	32	$-1.3340 \cdot 10^{10}$	1.8
C13	–	–	–	15	$-6.5791 \cdot 10^{7}$	0.2
C14	–	–	–	24	$-6.5593 \cdot 10^{9}$	1.9
C15	8	-0.7362	0.6	25	-0.7500	0.7
C16	–	–	–	10	$-7.2000 \cdot 10^{7}$	0.1
C17	–	–	–	9	$-1.2500 \cdot 10^{9}$	0.4
C18	1	$1.1755 \cdot 10^{3}$	0.2	–	–	–
C19	1	$2.2204 \cdot 10^{-16}$	0.2	–	–	–

Table 4 Results of some CUTEr problems and EROSEN

5.3 Test set 3: Separable quadratic programs

In the last two examples we apply MINQ8 and MINQ5 to separable quadratic problems with linear constraints by solving the dual problems. For comparison we only apply quadprog to the problems (the original, not the dual ones) since NewtonKKTqp requires a feasible starting point.

Example 1 We consider the minimization of a separable quadratic form subject to linear constraints

$$\begin{array}{l} \min\ c^T x + \frac{1}{2} x^T D x \\ \text{s.t.}\quad (Ax)_I \geq b_I, \quad (Ax)_E = b_E, \end{array} \tag{15}$$

where $c \in \mathbb{R}^n$, D is a positive definite $n \times n$ diagonal matrix, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and (I,E) is a partition of $\{1,\ldots,m\}$ with $|E| \leq n$. This problem is a special case of (10) and we proceed as described in Section 4.

We construct the instances in the following way. We first choose $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^n$, $y_{\text{des}}, s \in \mathbb{R}^m$ and the diagonal elements of D from a uniform distribution on $[0,1]$. Let $I = N \cup N'$, where N is the set of nonactive and N' the set of active inequalities, and $|N'| + |E| \leq \min(m,n)$. Then we set $y_{\text{des},N} = 0$, $s_{E \cup N'} = 0$, $x_{\text{des}} := D^{-1}(A^T y_{\text{des}} - c)$, and $b := Ax_{\text{des}} - s$. In the case $N = \emptyset$ (which can only happen if $m \leq n$), all indices are active or equalities and $b = Ax_{\text{des}}$.

For the solution vector x we define the vector $r \in \mathbb{R}^n$ by $r_E := (Ax - b)_E$ and $r_I := \min((Ax - b)_I, 0)$, i.e., r measures the violation of the constraints. Moreover, we use the abbreviation $\Delta x := x - x_{\text{des}}$. For MINQ5, we use the program `minqsep.m` included in the MINQ5 package. It applies MINQ5 to the dual problem, and one step of iterative refinement is applied if $|r| \leq \text{nnz}(A)\varepsilon(|A||x| + |b|)$ (with componentwise absolute value and inequalities) is violated, where nnz(A) denotes the number of nonzeros of the matrix A and ε the machine precision. For MINQ8, we solve the dual problem with $\texttt{tol} = \varepsilon = 2.2204 \cdot 10^{-16}$ (the machine precision) instead of the default value since the default value does not yet yield the optimal f_{best} for the original problems for the test cases with $I \neq \emptyset$. (Note that MINQ8 and MINQ5 are not applied to the original problems but to the dual ones.) We also apply one step of iterative refinement in the above-mentioned case.

Table 5 contains the problem characteristics of the problems of Example 1. For each value of (m,n), we consider one instance with empty N' and one with nonempty N'. In Table 6 the test results for MINQ8, MINQ5 and the interior-point-convex `quadprog` algorithm are presented; two problems cannot be solved with `quadprog` due to memory problems.

No.	m	n	$\lvert E\rvert$	$\lvert N'\rvert$
S1	700	1000	500	10
S2	700	1000	700	0
S3	2000	3000	1000	300
S4	2000	3000	2000	0
S5	7000	10000	5000	100
S6	7000	10000	7000	0
S7	1000	1000	500	100
S8	1000	1000	1000	0
S9	3000	3000	1500	300
S10	3000	3000	3000	0
S11	1000	700	500	100
S12	1000	700	700	0
S13	3000	2000	1500	300
S14	3000	2000	2000	0

Table 5 Problem characteristics of Example 1

	MINQ8				MINQ5			quadprog (ipc)		
No.	nit	$\|r\|_\infty$	$\|\Delta x\|_\infty$	t[s]	$\|r\|_\infty$	$\|\Delta x\|_\infty$	t[s]	$\|r\|_\infty$	$\|\Delta x\|_\infty$	t[s]
S1	40	$1.98\cdot10^{-9}$	$4.05\cdot10^{-10}$	0.9	$4.07\cdot10^{-9}$	$9.90\cdot10^{-10}$	7.2	$2.56\cdot10^{-9}$	$7.22\cdot10^{-4}$	39
S2	3	$1.08\cdot10^{-6}$	$3.03\cdot10^{-7}$	0.1	$6.29\cdot10^{-9}$	$2.39\cdot10^{-9}$	2.4	$6.52\cdot10^{-9}$	$1.43\cdot10^{-9}$	3.5
S3	307	$5.59\cdot10^{-8}$	$1.30\cdot10^{-3}$	54	$5.22\cdot10^{-8}$	$1.21\cdot10^{-8}$	53	$7.08\cdot10^{-8}$	$3.50\cdot10^{-3}$	524
S4	3	$3.91\cdot10^{-8}$	$5.44\cdot10^{-9}$	1.6	$4.10\cdot10^{-8}$	$6.46\cdot10^{-9}$	47	$5.40\cdot10^{-8}$	$5.48\cdot10^{-9}$	49
S5	533	$1.25\cdot10^{-6}$	$4.70\cdot10^{-3}$	2183	$1.06\cdot10^{-6}$	$7.82\cdot10^{-8}$	9938	–	–	–
S6	2	$3.10\cdot10^{-6}$	$2.40\cdot10^{-7}$	42	$1.13\cdot10^{-6}$	$1.16\cdot10^{-7}$	3921	–	–	–
S7	97	$7.92\cdot10^{-9}$	$1.23\cdot10^{-9}$	2.6	$7.92\cdot10^{-9}$	$8.73\cdot10^{-10}$	33	$8.15\cdot10^{-9}$	$9.82\cdot10^{-4}$	67
S8	2	$2.76\cdot10^{-7}$	$1.62\cdot10^{-6}$	0.2	$5.36\cdot10^{-9}$	$3.00\cdot10^{-8}$	6.9	$4.89\cdot10^{-9}$	$2.74\cdot10^{-8}$	4.1
S9	772	$3.35\cdot10^{-8}$	$5.12\cdot10^{-9}$	246	$5.03\cdot10^{-8}$	$2.24\cdot10^{-8}$	761	$4.10\cdot10^{-8}$	$2.30\cdot10^{-3}$	1165
S10	3	$1.02\cdot10^{-7}$	$1.97\cdot10^{-7}$	3.8	$8.38\cdot10^{-8}$	$1.83\cdot10^{-7}$	144	$9.31\cdot10^{-8}$	$3.28\cdot10^{-7}$	50
S11	169	$1.22\cdot10^{-9}$	$3.17\cdot10^{-10}$	4.4	$1.51\cdot10^{-9}$	$3.81\cdot10^{-10}$	35	$1.46\cdot10^{-9}$	$6.56\cdot10^{-4}$	33
S12	3	$3.73\cdot10^{-7}$	$2.44\cdot10^{-6}$	0.1	$5.12\cdot10^{-9}$	$1.75\cdot10^{-8}$	12	$3.03\cdot10^{-9}$	$4.60\cdot10^{-9}$	4.8
S13	121	$5.40\cdot10^{-8}$	$7.14\cdot10^{-4}$	33	$5.49\cdot10^{-8}$	$5.98\cdot10^{-9}$	59	$4.75\cdot10^{-8}$	$2.18\cdot10^{-4}$	529
S14	3	$6.23\cdot10^{-7}$	$2.49\cdot10^{-6}$	2.0	$1.26\cdot10^{-7}$	$2.12\cdot10^{-7}$	255	$2.33\cdot10^{-8}$	$1.21\cdot10^{-7}$	79

Table 6 Results for Example 1

All three algorithms algorithms find approximately feasible points with the same function value up to at least 5 significant digits (not displayed here). Most of the points are very close to x_{des}, with the exception of a few solutions produced by MINQ8 and `quadprog` (usually the problems that converge most slowly). The active-set `quadprog` algorithm is faster than the interior-point-convex algorithm (but not faster than MINQ8) for 9 of the 12 problems that are solved by `quadprog`, but it is takes a long time to solve S9 and also S3 and therefore we do not display the results. The active-set `quadprog` algorithm handles problems with only bounds, or only linear equality constraints, but not both and is therefore not applicable.

Example 2 We consider the problem of finding a point of minimal norm satisfying a given system of linear inequalities,

$$\begin{aligned} &\min \ \tfrac{1}{2}\|x\|_2^2 \\ &\text{s.t.} \ \ Ax \geq b. \end{aligned} \tag{16}$$

This problem is a special case of (15) with $c = 0$, $D = I_n$, and $E = \emptyset$, and we consider the three cases $m < n$, $m = n$ and $m > n$. We applied MINQ8, MINQ5, and the active-set `quadprog` algorithm and present the results in Table 7. (The interior-point-convex algorithm yielded suboptimal feasible points for all problems.) MINQ8, MINQ5, and the active-set `quadprog` algorithm produce essentially the same solutions. We display the maximal violation $\|r\|_\infty$ of the constraints and the CPU time needed for the computation for all three algorithms and the number of iterations for MINQ8. The last block of Table 7 contains the minimal function value $\frac{1}{2}\|x\|_2^2$ of the solution and the number of constraints that are numerically active ($|(Ax)_i - b_i| \leq 10^{-8}$); these quantities are the same (up to the precision presented here) for the successful algorithms. `quadprog` does not solve the three largest problems due to memory problems.

		MINQ8			MINQ5		quadprog (as)		Solution	
m	n	nit	$\|\|r\|\|_\infty$	t[s]	$\|\|r\|\|_\infty$	t[s]	$\|\|r\|\|_\infty$	t[s]	$\frac{1}{2}\|\|x\|\|_2^2$	nact
700	1000	18	$9.24 \cdot 10^{-14}$	0.3	$4.11 \cdot 10^{-15}$	0.2	$2.33 \cdot 10^{-15}$	4.6	$1.9467 \cdot 10^{-3}$	17
2000	3000	43	0	2.0	$4.33 \cdot 10^{-15}$	1.3	$2.55 \cdot 10^{-15}$	150	$6.6330 \cdot 10^{-4}$	25
7000	10000	80	$3.30 \cdot 10^{-14}$	50	$7.33 \cdot 10^{-15}$	30	–	–	$1.9857 \cdot 10^{-4}$	55
1000	1000	27	$2.16 \cdot 10^{-13}$	0.3	$3.55 \cdot 10^{-15}$	0.3	$2.33 \cdot 10^{-15}$	4.3	$1.9656 \cdot 10^{-3}$	18
3000	3000	42	$4.11 \cdot 10^{-15}$	3.6	$4.00 \cdot 10^{-15}$	2.8	$2.66 \cdot 10^{-15}$	188	$6.5571 \cdot 10^{-4}$	38
10000	10000	85	$2.15 \cdot 10^{-14}$	84	$6.55 \cdot 10^{-15}$	62	–	–	$1.9857 \cdot 10^{-4}$	62
1000	700	26	$7.05 \cdot 10^{-14}$	0.2	$3.77 \cdot 10^{-15}$	0.2	$2.78 \cdot 10^{-15}$	2.0	$2.8136 \cdot 10^{-3}$	16
3000	2000	30	$3.91 \cdot 10^{-13}$	2.3	$3.44 \cdot 10^{-15}$	2.5	$2.55 \cdot 10^{-15}$	49	$9.9814 \cdot 10^{-4}$	31
10000	7000	99	$1.24 \cdot 10^{-14}$	63	$6.11 \cdot 10^{-15}$	46	–	–	$2.8584 \cdot 10^{-4}$	62

Table 7 Results for Example 2

5.4 Discussion summary

Since MINQ8 gains a first or second place on all of our test sets, MINQ8 can be regarded as the best of the tested algorithms on our test set. For many of the problems there is one `quadprog` algorithm that performs well, but there is no easy general rule for choosing the "right" `quadprog` algorithm. We used the interior-point-convex algorithm for the convex and the trust-region-reflective algorithm for the non-convex problems on the first two test sets, but for Test Set 5.2 we took the trust-region-reflective algorithm for all problems with bound constraints and not just the non-convex ones, for Example 1 (convex problems) we present the results with the interior-point-convex problems (consistent with the choice on the Test Set 5.1), but for Example 2 we chose the active-set algorithm. `NewtonKKTqp` is slow and fails to solve many problems due to memory problems. MINQ8 and MINQ5 are the only of the tested algorithms that are applicable to all problems of our test sets and MINQ8 is the winner due to its better performance on Test Set 5.1, Test Set 5.2 except for NCVXBQP2, and Example 1.

References

1. Absil, P.-A., Tits, A.L.: Newton-KKT interior-point methods for indefinite quadratic programming. Comput. Optim. Applic. 36, 5–41 (2007). `http://www.montefiore.ulg.ac.be/~absil/Publi/indefQP.htm`
2. Dietz, T., Verl, A.: Simulation of the stopping behavior of industrial robots using a complementarity-based approach. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 428–433. IEEE (2011)
3. Fernandes, R., Fraser, R., Latifovic, R., Cihlar, J., Beaubien, J., Du, Y,: Approaches to fractional land cover and continuous field mapping: A comparative assessment over the BOREAS study region, Remote Sensing of Environment 89, 234–251 (2004)
4. Floudas, C.A., Pardalos, P.M.: A Collection of Test Problems for Constrained Global Optimization Algorithms. Springer, Berlin (1990)
5. Fourer, R.: Staircase matrices and systems, SIAM Review 26, 1–70 (1984)
6. Friedlander, M.P., Leyffer, S.: Global and finite termination of a two-phase augmented Lagrangian filter method for general quadratic programs. SIAM J. Sci. Comput. 30, 1706–1729 (2008)
7. Gao, Y., Choudhary, A., Hua, G.: A nonnegative sparsity induced similarity measure with application to cluster analysis of spam images. In: 35th IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 5594–5597. IEEE (2010)

8. Gould, N.I.M., Orban, D., Toint, Ph.L.: CUTEr (and SifDec), a constrained and unconstrained testing environment, revisited. CERFACS Technical Report No. TR/PA/01/04 (2004)
9. Gould, N.I.M., Toint, Ph.L.: A quadratic programming page. `http://www.numerical.rl.ac.uk/people/nimg/qp/qp.html`. Accessed 21 July 2015
10. Haruno, M., Wolpert, D.M.: Optimal control of redundant muscles in step-tracking wrist movements, J. Neorophysiol. 94, 4244–4255. (2005)
11. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization. 2nd Edition. Kluwer, Dordrecht (2000)
12. Huyer, W., Neumaier, A.: Global optimization by multilevel coordinate search, J. Global Optimization 14, 331–355 (1999)
13. Huyer, W., Neumaier, A.: SNOBFIT – stable noisy optimization by branch and fit, ACM Trans. Math. Software 35, No. 2, Article 9 (2008)
14. Kannan, A., Wild, S.M.: Obtaining quadratic models of noisy functions, Preprint ANL/MCS-P1975-1111, Argonne National Laboratory (2012)
15. Kanzow, C., Petra, S.: Projected filter trust region methods for a semismooth least squares formulation of mixed complementarity problems, Optimization Methods and Software 22, No. 5, 713–735 (2007)
16. Kybic, J.: High-dimensional entropy estimation for finite accuracy data: *R*-NN entropy estimator. In: Karssemeijer, N., Lelieveldt, B. (eds.) Information Processing in Medical Imaging. Lecture Notes in Computer Science Vol. 4584, pp. 569–580. Springer, Berlin (2007)
17. Lee, J., Leyffer, S. (eds.): Mixed Integer Nonlinear Programming. IMA Volumes in Mathematics and its Applications, Vol. 154, Springer, Berlin (2012)
18. Lin, Y.Y., Pang, J.-S.: Iterative methods for large convex quadratic programs: a survey. SIAM J. Control and Optimization 25, 383–411 (1987)www
19. Liu, W., Brugger, J., McPhail, D.C., Spiccia, L.: A spectrophotometric study of aqueous copper(I)-chloride complexes in LiCl solutions between 100 °C and 250 °C, Geochimica et Cosmochimica Acta 66, No. 20, 3615–3633 (2002)
20. Neumaier, A.: MINQ: General definite and bound constrained indefinite quadratic programming, Web document, `http://www.mat.univie.ac.at/~neum/software/minq/` (1998)
21. Olshansky, Y., Turkel, E.: Simultaneous scatterer shape estimation and partial aperture far-field pattern denoising, Commun. Comput. Phys. 11, No. 2, 271–284 (2012)
22. Roos, M., Rothe, J., Scheuermann, B.: How to calibrate the scores of biased reviewers by quadratic programming. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, pp. 255–260. AAAI (2011)
23. Shanno, D.F.: On variable-metric methods for sparse Hessians, Math. Comput. 34, 499–514 (1980)
24. Smit, W.J., Barnard, E.: Continuous speech recognition with sparse coding, Computer Speech and Language 23, 200–219 (2009)
25. Voglis, C., Lagaris, I.E.: BOXCQP: An algorithm for bound constrained convex quadratic problems. In: 1st International Conference "From Scientific Computing to Computational Engineering". IC-SCCE, 2004.
26. Xu, S.: A non-interior path following method for convex quadratic programming problems with bound constraints. Computat. Optim. Appl. 27, 285–303 (2004)